4NL3 Assignment 3

Guneev Arora - 400477579

March 3, 2025

Contents

1 Embeddings

1.1 Methodology

In this section I Discuss the approach I took to normalizing my data for the models. Furthermore, what are the options that the user can select when performing the text normalization steps.

1.2 Tokenization & Normalization

A Lot of this was taken straight from my first & second assignment
To begin with, I started with tokenization which I did with a simple python for loop through the data and also used the nltk libary for some of the tools.

Now in terms of the normalization tools, I created these normalization tools in my last assignment and will use the same ones:

1. Lowercase: Makes all the text lowercase.
2. Stemming: Gets rid of the suffixes of the words in the data.
3. Lemmatize Words: Lemmatizes the words in the data.
4. Remove Stop words: Removes common words that do not mean much such as "and".
5. Remove Punctuation and single letters: Removes all the separate punctuation from the data along with single letters.
6. Remove Numbers: This is mainly just to remove all numbers for some filtering use I will explain later.

Based on a few tests, I decided to lowercase, remove punctuation and single letters, remove stop words and remove numbers.

1.3 Two variations of Embeddings

Now that we have our data ready, we need to make our models, I decided to make 2 models, one with skip-gram and the other with CBOW. I will now explain my parameters for these models. Both models are identical other than the model type.

To make these models I used the gensim models library to import Word2Vec

Model settings:

1. Window Size: 10
2. vector size: 300
3. Min Count: 5
4. Model Type: skip-gram or CBOW

I decided to use different models as my test case as the way they work is different as skip-gram uses the given word to predict the context words and CBOW uses the context words to predict the context ,while the CBOW model tries to use the context to predict the word which I think is very cool and something I would want to test.

As to why I used my settings:

1. Window Size: I decided to use size 10 as I thought that for this assignment, it would help make my model a lot more generalizable as it can get and understand relationships between words more in depth.
2. Vector Size: I decided to use 300, as I saw some other data later also using 300 dimensions and thought that it would be the best as it wouldn't cause much overfitting while also keeping the meaning between words while keeping it somewhat consistent.
3. Min Count: Based on some research, 5 seemed like the best value to use to keep data pretty accurate while cutting down some noise.

In terms of what I learned, I mentioned a lot of it for my reasoning on using each model, but on top of that I had to do research on when each model is better to use, and why it's better. As mentioned, I learned why each parameter affects the data and impacts the results while also learning how to apply them.

1.4 Five Queries

To begin with, the two pretrained models I used were the Glove-wiki-gigaword-300 and the fasttext-wiki-news-subwords-300, keeping the 300 dimensions pretty consistent. I chose these due to size and ease of use, as importing them using import Gensim was east using the gensim downloader libary.

Now here are my five queries:

1. Query 1: april
2. Query 2: bird + song
3. Query 3: france + year - paris
4. Query 4: computer + hardware - laptop
5. Query 5: julian - germany

I chose these, as I thought I could get some cool and useful results using them as well as getting to test other random results when if it wasn't as accurate as I would have hoped for.

Now here are my results:

Query	Model 1	Model 2	Pretrained Model 1	Pretrained Model 2
april	february (0.922)	march (0.874)	june (0.959)	february (0.832)
	march (0.920)	february (0.799)	october (0.956)	june (0.825)
	october (0.910)	june (0.779)	july (0.955)	july (0.818)
	september (0.903)	october (0.757)	february (0.953)	october (0.815)
	december (0.898)	november (0.749)	september (0.940)	january (0.800)
	june (0.897)	september (0.729)	november (0.937)	september (0.780)
	july (0.894)	december (0.724)	december (0.937)	december (0.776)
	november (0.888)	july (0.722)	january (0.932)	febuary (0.773)
	august (0.886)	january (0.710)	march (0.929)	november (0.763)
	january (0.813)	august (0.696)	august (0.928)	feb (0.757)
bird + song	startin (0.599)	ballad (0.543)	songs (0.632)	songbird (0.753)
	songs (0.599)	birds (0.542)	album (0.597)	songster (0.737)
	zebrahead (0.590)	bee (0.516)	birds (0.578)	birdy (0.732)
	pnb (0.587)	marabou (0.502)	recorded (0.557)	blackbird (0.716)
	teardrops (0.581)	songs (0.501)	sings (0.527)	bird-song (0.710)
	tisket (0.579)	caged (0.499)	flu (0.522)	skylark (0.706)
	oughta (0.577)	dove (0.488)	singing (0.507)	lyrebird (0.694)
	mooo (0.575)	sings (0.483)	singer (0.486)	birds (0.694)
	mmmbop (0.572)	lovin (0.475)	tune (0.478)	humming-bird (0.687)
	tasket (0.570)	melody (0.473)	theme (0.466)	half-bird (0.681)
france + year - paris	years (0.494)	week (0.388)	last (0.628)	month (0.661)
	three (0.438)	years (0.382)	years (0.612)	decade (0.630)
	time (0.421)	decade (0.351)	month (0.609)	year–in (0.627)
	second (0.418)	weeks (0.350)	months (0.593)	years (0.615)
	since (0.410)	months (0.349)	since (0.572)	week (0.605)
	four (0.403)	winnings (0.333)	ago (0.536)	year– (0.601)
	two (0.402)	eligible (0.317)	decade (0.529)	year–and (0.600)
	became (0.391)	month (0.315)	already (0.520)	year-and (0.597)
	history (0.388)	worlder (0.314)	third (0.516)	years-and (0.596)
	named (0.378)	dominion (0.310)	second (0.516)	year-year (0.593)
computer + hardware - laptop	software (0.541)	computers (0.637)	software (0.677)	software (0.751)
	computers (0.538)	software (0.598)	systems (0.585)	computer-software (0.689)
	computing (0.519)	cpu (0.560)	technology (0.531)	hardware-software (0.677)
	programmer (0.499)	computing (0.560)	computing (0.504)	computer-programming (0.667)
	microprocessor (0.496)	malware (0.560)	engineering (0.491)	computer-system (0.666)
	algorithmic (0.489)	memory (0.560)	computers (0.481)	hardware-related (0.665)
	compilers (0.486)	user (0.552)	components (0.475)	data-processing (0.664)
	minicomputers (0.484)	programmers (0.551)	simulation (0.475)	hardware-oriented (0.653)
	oisc (0.484)	programmer (0.550)	graphics (0.472)	hardwares (0.651)
	opcodes (0.482)	instructions (0.547)	equipment (0.470)	non-computer (0.648)
julian - germany	calendar (0.382)	kay (0.531)	tavarez (0.447)	juliana (0.347)
	gregorian (0.349)	gemma (0.527)	casablancas (0.420)	julie (0.336)
	leap (0.324)	darren (0.503)	guyer (0.420)	julia (0.327)
	tuesday (0.288)	andre (0.500)	brendon (0.414)	Hijri (0.325)
	domini (0.282)	neal (0.499)	tuwim (0.399)	juliet (0.322)
	lentulus (0.271)	rathbone (0.499)	fellowes (0.392)	dhu (0.322)
	starting (0.269)	silas (0.497)	colbeck (0.391)	juliae (0.318)
	wednesday (0.267)	adrian (0.490)	terrill (0.380)	Gregorian (0.316)
	consulship (0.259)	gavin (0.486)	callow (0.371)	julio (0.311)
	sunday (0.254)	daniels (0.484)	argüelles (0.363)	Julian (0.311)

Table 1: Query Results for Different Models

In terms of the difference in data:

The probabilities are very different for all of them as I said before, some models believe in the results more, and usually the skip-gram has higher probabilities. For

topics that are a little open and not too similar, we can see that the results are very different between is very clear as most words are different for these.

In terms of the similarity in data:

Most words have something in similar on all models, the results are not too far apart for most of it which is good, but of course with many different probabilities and order.

In terms of what interested and surprised me:

It was really interesting to see how topics that don't have much collection such as bird and song have very different results for each model as all of them take it as something different. This made me do some more research on how it happened and why it happens which led to a lot of random learning.

I was also very surprised to see how similar the results can be for some topics, as I had originally thought there would be at least a small difference in every model, but some models were even closer than I could have ever thought.

2 Bias

2.1 Methodology

For the Bias part of the assignment, I decided to use the WEAT based study with target terms male-names and weapons along with the pleasant and unpleasant words for the target.

In terms of the implementation, I used query format from the examples to make my own query, and to run them, I used the code provided in the assignment document to assist me keeping the same format and settings. Once we had everything set up, it ran the queries and gave the bias result for each of the 4 models (my 2 models and 2 pretrained models)

In terms of what I observed, I had a lot of words missing within my models which was pretty annoying to deal with as there was no data for those queries , however for the results I could obtain, everything was pretty consistent but the CBOW usually had a higher bias than all the others by a good amount, however there were some

places where the values of biases were very different such as European and African names with Pleasant and Unpleasant.

2.2 Query & Results

	skipgram	cbow	glove-wiki-gigaword-300	fasttext-wiki-news-subwords-300	word2vec-google-news-300 (WEAT original)	glove-wiki-gigaword-300 (WEAT original)
Flowers and Insects wrt Pleasant(5) and Unpleasant(5)	0.98	1.02	1.27	1.36	1.54	1.50
Instruments and Weapons wrt Pleasant(5) and Unpleasant(5)	1.73	3.04	1.91	1.65	1.63	1.53
European american names(5) and African american names(5) wrt Pleasant(5) and Unpleasant(5)	-0.13	1.12	1.28	0.43	0.58	1.41
European american names(7) and African american names(7) wrt Pleasant(5) and Unpleasant(5)	NaN	NaN	0.85	0.14	1.24	1.50
European american names(7) and African american names(7) wrt Pleasant(9) and Unpleasant(9)	NaN	NaN	1.07	0.30	0.72	1.28
Male names and Female names wrt Career and Family	1.02	2.99	1.32	0.44	1.89	1.81
Math and Arts wrt Male terms and Female terms	NaN	NaN	0.24	0.20	0.97	1.06
Science and Arts 2 wrt Male terms and Female terms	NaN	NaN	0.32	0.11	1.24	1.24
Mental disease and Physical disease wrt Temporary and Permanent	0.19	0.12	0.57	0.20	1.30	1.38
Young peoples names and Old peoples names wrt Pleasant(9) and Unpleasant(9)	-0.03	-0.15	0.21	-0.03	-0.08	1.21

Figure 1: Original WEAT querys and result

Query	skip-gram	CBOW	GloVe-wiki-gigaword-300	fastText-wiki-news-subwords-300
Male names and female names wrt weapons and family	0.63	2.26	0.64	0.44

Table 2: My new added WEAT query and result

2.3 Consequences

In terms of the consequences of using these embeddings might be when using them as features in a machine learning model, I think that the bias can cause super biased

and unfair results another, just due to stereotyping of racial, sexist or other topics caused by this. One example is the query I added that shows that male names and weapons have a higher bias.

These could cause a lot of ethical problems for companies and people making models as it could come off in the wrong way or produce models that will and can cause problems socially.

3 Classification

3.1 Methodology

This part was easily the trickiest part of the assignment and where I struggled the most. Using help from the tutorial and chatgpt, I was able to finally get a model that uses logical regression to classify the data.

3.2 Bag Of Words Model

For this classification, I used my data from the previous assignment (TV show subtitles) along with a new data filtering method to format the files as needed as I had converted the files to text documents last time.

I used some regex along with normal for loops to get the data in the format I wanted, and then used the train test split function from sklearn to split the data properly (I had to use zip to format data properly which was suggested by chatgpt as I had some problems before I did this) and then turned it into a bag of words.

Now, we could easily turn it into a LogisticRegression model as done in the tutorial to calculate the accuracy and F1 score I will show after explain the other model.

3.3 Embeddings Model

Now, for this model, I decided to use the skipgram model as I thought it would add more verity and give me more to test. I filtered out words that dont exist in the dictionary to make sure no errors are caused and then let

Then we do the same and run the LogisticRegression again the same way to get the values again.

3.4 Results

Here are my average and F1 scores for each of the models.

Bag of Words Model:

1. Accuracy: 0.771777401801637
2. F1 Score: 0.8238947108429483

Embeddings Model:

1. Accuracy: 0.6068864602862499
2. F1 Score: 0.7484123964765445

So, as you can see in terms of accuracy obviously the bag of words model is a lot more accurate and has a higher F1 score.

This also does make sense as the bag of words uses specially new data from the 2 shows while the embeddings model gets to use general English which can be unique but thankfully still has different values due to the nature of the shows. As one show uses simple English and basic humor, while the other is fantasy based and had a lot of random words that are very unique.

In terms of efficiency, as the bag of words model is a lot faster in general, for this classification case is just better overall.

4 Reflection

4.1 What I learned

I learned a lot in this assignment, specially about biases and how metrics like WEAT work. Along with that I also got to learn a lot on how to make and use different models along with how to compare them.

I also got to learn a lot on the consequences of using the embeddings and how they can affect results.

Lastly I got to learn a lot on classification and how to use logistic regression to get results and compare data.

4.2 What was unexpected

I think some of the biases were decently unexpected as I did not realize how much of a difference it actually makes. Furthermore, I did see a little inconsistency with the bias sporadically which was pretty surprising, mainly caused by the skipgram model.

4.3 Challenges & how I overcame them

For me the most challenging part was part 4.4 of this question, as I had trouble figuring it out, and had to use chatgpt a lot on what I have to do and on how to do it, along with that the tutorial 4 files helped me a lot as well and I was able to get the code in the end, however it did require a lot of help.

5 AI Tool Usage

For AI tools, I used ChatGPT, I did around 62 queries which each are 4.26g = 264.12 grams of CO2 total with all of them being queries on some coding questions but not directly copying code and some other questions on certain topics.